Hyperloop
Route Optimization

Finding the path of least time and least cost using techniques of Mathematical Analysis

By Jonathan Ward & David Roberts

Harvard University Hyperloop Research Team (HUHRT)

Preface

This document records a simple project which I and one of my classmates took on over several weeks of our Winter Break, just on a whim. We're both Math/Physics concentrators, but we take a special interest in potentially transformative technologies. Elon Musk's Hyperloop transportation system, with its promise for cheap, fast, and green transportation, falls into that category. For more details on the proposal, check it out online via Google Search.

There were many aspects of the Hyperloop project that we could have worked on, but I felt that route optimization was an area that could play to our prior experience with functional analysis on abstract spaces, a course which both of us thoroughly enjoyed, because, I must say a cliche, the astonishingly powerful and beatiful ideas expanded our minds.

Therefore our journey begins at the most abstract aspect of the Hyperloop project: determining the route. What did Musk have to say about this? Well, in his Alpha document describing the Hyperloop proposal, Elon Musk stated that he was looking for a trip path and velocity that did not exceed .5 g's at any point. Our goal was to minimize trip time and cost under this passenger comfort constraint, using the path and velocity profile as our variables.

The results are, to say the least, interesting. Over the course of just a few weeks, Jon and I learned an incredible amount in such diverse topics, from computer science, to mathematics, to physics. We both hope you find beauty as well, as you navigate this document.

Enjoy!

Chapter 1

Calculating Time T($\boldsymbol{\gamma}$)

1.1 The G-force Constraint and Resulting Equation

Presume a proposed Hyperloop path $\boldsymbol{\gamma} : \mathbb{R} \to \mathbb{R}^3$ is parametrized by the distance along the path. We can then express $\boldsymbol{\gamma}$ in an Earth-Centered, Earth-Fixed coordinate system:

$$\boldsymbol{\gamma}(s) = (x(s), y(s), z(s))$$

Then we have that the velocity profile $v(s)$, bounded by the maximum centripetal acceleration tolerance g, must satisfy the following inequality, with g_{rad} and g_{lin} being the centripetal and linear accelerations respectively:

$$g \leq \sqrt{g_{\text{rad}}^2 + g_{\text{lin}}^2} \qquad (g_{\text{lin}} = \tfrac{dv(s)}{dt}, \quad g_{\text{rad}} = \tfrac{v(s)^2}{r})$$

To find the optimal velocity profile we take the effective acceleration to be the maximum allowed value, which is the constant G, so that we have the equality case in equation (1).

$$\begin{aligned} G^2 = g_{\text{rad}}^2 + g_{\text{lin}}^2 &= \frac{v(s)^4}{r(s)^2} + \left(\frac{dv(s)}{dt}\right)^2 \\ &= \boxed{\frac{v(s)^4}{r(s)^2} + v(s)^2 \left(\frac{dv(s)}{ds}\right)^2} \qquad (s = vt) \end{aligned}$$

Therefore our g-force constraint yields a differential equation, whose solution, $v(s)$, is the optimal velocity profile we are looking for. The radius of curvature is a function of the path in the following way:

$$r(s) = \frac{(x'^2 + y'^2 + z'^2)^{\frac{3}{2}}}{\sqrt{(z''y' - y''z')^2 + (x''z' - z''x')^2 + (y''x' - x''y')^2}}$$

1.2 Solving the Equation Locally

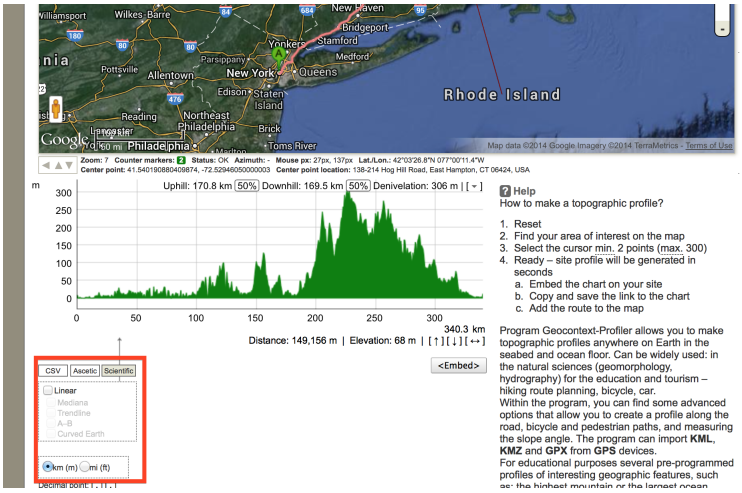
The boxed differential equation of the previous section is incredibly difficult to solve, because $r(s)$ is completely arbitrary, even chaotic. We therefore solve the simplest case, where $r(s)$ is constant, and break up the path into a series of regions of constant curvature.

1.2.1 Calculating Radii of Curvature

The first step, of course, is to convert a path into a series of sections of constant curvature. The process has three steps, and involves the internet, Google Earth, and Microsoft Excel:

(a) In Google Earth, get driving directions from point A to point B (This will be our path $\boldsymbol{\gamma}$), and export the route as a .kml file.

(b) Visit `http://www.geocontext.org/publ/2010/04/profiler/en/?import=kml`, and import your .kml file, and click the .csv option:



(c) Copy and paste the results into Microsoft Excel, and after pasting, click the clipboard for paste options, to allow Excel to recognize the .csv format of the contents. Now save the file as a .csv.

(d) This gives us a series of longitude and lattitude points with corresponding elevation (meters above sea level) $\{(\theta_i, \phi_i, e_i)\}_{i=1}^n$. These coordinates denote points on our path $\boldsymbol{\gamma}$. We then transform into an Earth-Centered, Earth-Fixed coordinate system, in order to calculate curvature.

$$\mathbf{r}_i = (x_i, y_i, z_i) = (R_\oplus + e_i)(\cos\theta_i \cos\phi_i, \cos\theta_i \sin\phi_i, \sin\theta_i) \in \mathbb{R}^3$$
$$(R_\oplus \equiv \text{Radius of the Earth})$$

(e) We then approximate the radius of curvature $R(\mathbf{r}_i)$ at the point $\mathbf{r}_i$ by the radius of the circle containing $\mathbf{r}_{i-1}, \mathbf{r}_i, \mathbf{r}_{i+1}$. This is a nonlinear system of three distance equations in 3D-space,

$$R(\mathbf{r}_i) = \|\mathbf{r}_{i-1} - \mathbf{c}\| = \|\mathbf{r}_i - \mathbf{c}\| = \|\mathbf{r}_{i+1} - \mathbf{c}\|$$

where $\mathbf{c} = (x, y, z)$ is the position of the center of the circle. There is now one last step: we want to describe the path as a series of *regions* of constant curvature, but we have

defined curvature for only the *points*. So for the interval $[s_i, s_{i+1}]$, we assign a radius of curvature as follows[1]:

$$R([s_i, s_{i+1}]) = \begin{cases} R(\mathbf{r}_i) & i = 1 \text{ or } n-1 \\ R(\mathbf{r}_{i-1}) + R(\mathbf{r}_i)]/2 & \text{otherwise} \end{cases}$$

1.2.2 Solving the Equation with Constant Curvature

We have made the discrete approximation of $r(s)$ precisely so that we can solve our differential equation with $r(s) = R$, a constant. Our solution then describes the optimal speed of the pod entering a circular segment with intitial velocity v_0:

$$\begin{aligned} G^2 &= \frac{v(s)^4}{R^2} + v(s)^2 \left(\frac{dv(s)}{ds}\right)^2 \\ &= \frac{1}{4}\left(\frac{du}{ds}\right)^2 + \frac{u(s)^2}{R^2} && (u = v^2) \\ \left(\frac{du}{ds}\right)^2 &= 4G^2\left[1 - \left(\frac{u(s)}{RG}\right)^2\right] && \text{(Roberts-Ward Equation)} \\ \int ds &= \frac{1}{2G}\int \frac{du}{\sqrt{1 - \frac{u^2}{R^2G^2}}} + C \\ &= \frac{R}{2}\int \frac{dx}{\sqrt{1-x^2}} + C && (x = u/RG) \\ s &= \frac{R}{2}\sin^{-1}\left(\frac{u}{RG}\right) + C \\ u(s) &= RG\sin\left(\frac{2}{R}(s - C)\right) \end{aligned}$$

We now solve for C based on the initial condition $u(0) = u_0$:

$$\begin{aligned} u_0 = -RG\sin\left(\frac{2C}{R}\right) \rightarrow\ C &= \frac{R}{2}\sin^{-1}\left(-\frac{u_0}{RG}\right) \\ u(s) &= RG\sin\left[\frac{2s}{R} + \sin^{-1}\left(\frac{u_0}{RG}\right)\right] \\ \boxed{v(s) = \sqrt{RG\sin\left[\frac{2s}{R} + \sin^{-1}\left(\frac{v_0^2}{RG}\right)\right]}} && \text{(Roberts-Ward Solution)} \end{aligned}$$

Note that this equation does not account for a maximum velocity prescribed by any other considerations such as thermodynamic and stress considerations, shockwaves. Furthermore this equation assumes that the velocity can be modified at will, in particular that there are no restrictions on the placement of the linear accelerators and also that the acceleration is smooth.

[1]Note that this formula is symmetric, forwards and backwards. This will turn out to be immensely useful when solving the differential equation, because the physics of the solution contains this symmetry. In other words, the path velocity should be the same whether you go from start to finish (forwards), or from finish to start (backwards).

1.3 Solving the Equation Globally

The goal now is to stitch together local solutions (for $v(s)$) to form a global optimal velocity curve. The idea is to break up this job into two steps. We first solve for the maximum values of the velocity at the boundaries of each segment of the path, i.e. the entering and exiting velocities for each section. Note that we have to be careful, for if such velocities are too far apart, and the section is too short, the resulting solution will be discontinuous.

1.3.1 Finding Optimal Boundary Conditions

(a) Focus on u; in particular, focus on $\{u(s_i)\}_{i=1}^n \subset \mathbb{R}$, i.e. the values of u at the boundaries. Once these boundary conditions are solved for (which is the goal of this algorithm), the analysis of the next subsection should be able to determine $u(s)$ for all $s \in \mathbb{R}$.

(b) Set $\{u(s_i)\}_{i=1}^n$ initially to be $\{GR(s_i)\}_{i=1}^n$. Of course there does not exist a u satisfying these n boundary conditions, but we have to have a starting point that aims high. In particular, we know that the actual solution to u has boundary values less than these initial ones (unless the entire route has uniform curvature).

(c) We know that the algorithm has finished, (i.e. that a continuous u satisfying the boundary conditions exists) once, if for all i from 1 to n:

$$u_{i+1} \in [u_i + \Delta_+(u_i), u_i - \Delta_-(u_i)] \qquad \text{(Notation: } u_i \equiv u(s_i)\text{)}$$

Where $\Delta_+(u_i), \Delta_-(u_i)$ are respectively the maximum positive and negative changes in u permitted due to the physics of the region as a function of the initial square velocity u_i. More specifically,

$$\Delta_+(u_i) = \sup_{s\in[0,\Delta s_i]} \left(R_i G \sin\left[\frac{2s}{R} - \sin^{-1}\left(-\frac{u_i}{R_i G}\right)\right] - u_i \right)$$
$$\Delta_-(u_i) = \left| \inf_{s\in[0,\Delta s_i]} \left(R_i G \sin\left[-\frac{2s}{R} - \sin^{-1}\left(-\frac{u_i}{R_i G}\right)\right] - u_i \right) \right|$$

Where if $\Delta_-(u_i) \geq u_i$, as an extra step we set $\Delta_-(u_i) = u_i$, so as to not permit negative velocities and thus wasted time.

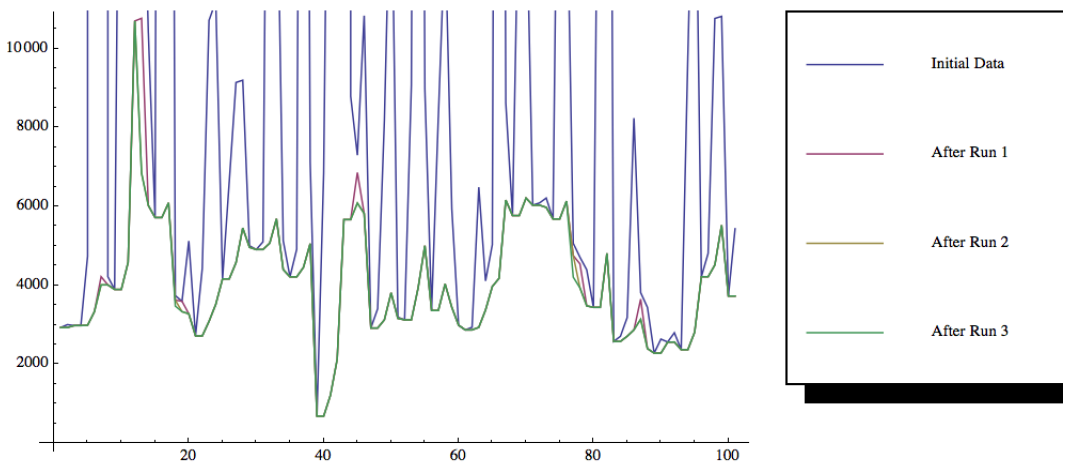
(d) Our algorithm thus is the following: if the algorithm is not finished, then there exists some i such that

$$u_{i+1} \notin [u_i - \Delta_-(u_i), u_i + \Delta_+(u_i)]$$

Now pick the least such i. Then execute the following:

If	Do
$u_{i+1} > u_i + \Delta_+(u_i)$	Set $u_{i+1} = u_i + \Delta_+(u_i)$
$u_{i+1} < u_i - \Delta_-(u_i)$	Set $u_i = u_{i+1} + \Delta_-(u_i)$

To see this algorithm in action, let's now focus on what happens to the first 100 regions, for detail. Notice how the algorithm is making incredibly minor changes by the 3rd run through:



For a driving route from SF to LA, the boundary conditions converge after three iterations of the algorithm.

1.3.2 Revisiting the Case of Constant Curvature

Consider a region of curvature R_i lasting from s_i to s_{i+1}. By the above algorithm, we determine the boundary conditions to be: $u(s_i) = u_i$, and $u(s_{i+1}) = u_{i+1}$. Our goal in this section is to find the optimal curve $u(s)$ connecting these two points.

According to the Roberts-Ward solution the transition u-value between a boundary point u_1 and a boundary point u_2 where s_{zero} is the s value of the transition point between u_1 and u_2, and G is a parametric value for the maximum allowed acceleration is given by

$$\text{SolFun}(u_1, u_2, s_{zero}, G, s) = u_2 \sin\left[\frac{2G}{u_2}((s - s_{zero}) + \frac{u_2}{2G}\arcsin\left(\frac{u_1}{u_2}\right))\right] \tag{1.1}$$

A physically realistic solution requires that the pod steadily maintain its maximum allowed u-value after achieving it. The maximum allowed u-value is attained when the argument of the sin function in $u(s)$ equals $\frac{\pi}{2}$

$$\begin{aligned}
\frac{2G}{u_2}\left((s_{\max} - s_{\text{zero}}) + \frac{u_2}{2G}\arcsin\left(\frac{u_1}{u_2}\right)\right) &= \frac{\pi}{2} \\
\left((s_{\max} - s_{\text{zero}}) + \frac{u_2}{2G}\arcsin\left(\frac{u_1}{u_2}\right)\right) &= \frac{u_2}{2G}\frac{\pi}{2} \\
(s_{\max} - s_{\text{zero}}) &= \frac{u_2}{2G}\frac{\pi}{2} - \frac{u_2}{2G}\arcsin\left(\frac{u_1}{u_2}\right) \\
(s_{\max} - s_{\text{zero}}) &= \frac{u_2}{2G}\left(\frac{\pi}{2} - \arcsin\left(\frac{u_1}{u_2}\right)\right) \\
s_{\max} &= \frac{u_2}{2G}\left(\frac{\pi}{2} - \arcsin\left(\frac{u_1}{u_2}\right)\right) + s_{\text{zero}}
\end{aligned}$$

Now we can define a piecewise transition function, TransFun.

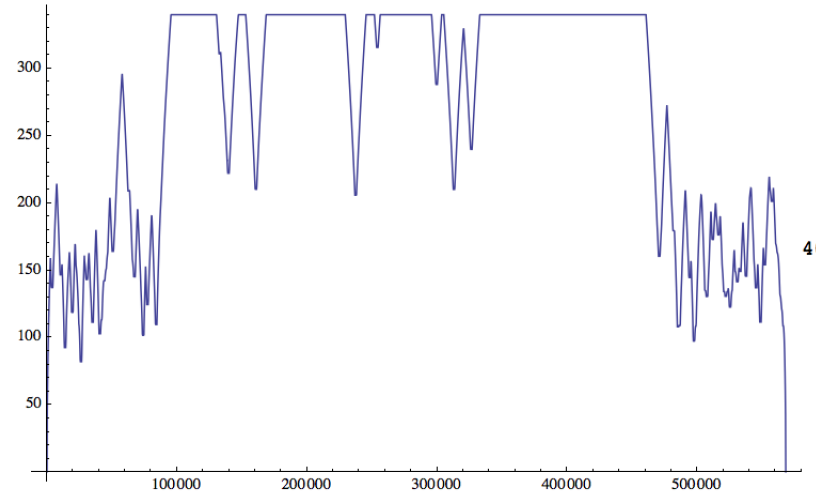
There a three cases for TransFun to handle: $u_1 > u_2$, $u_2 > u_1$, and $u_2 == u_1$. In the trivial case $u_1 == u_2$, the solution function is identically zero and Transfun returns u_1 as the u-value.
In the case where $u_1 < u_2$ we can make Transfun$(u_1, u_2, s_{zero}, G, s)$ physical over the entire domain by setting the u-value equal to u_1 in the region with $s < s_{zero}$, equal to SolFun$(u_1, u_2, s_{zero}, G, s)$ in the region with $s_{zero} < s < s_{max}$, and equal to u_2 in the region with $s_{max} < s$.

In the case that that $u_2 < u_1$ we can use the Roberts-Ward solution again by exploiting the symmetry of the physical system. If we reflect the boundary conditions across the line $s = 0$, then we obtain boundary conditions identical those in the case that $u_1 < u_2$. So by swapping the roles of u_1 and u_2 in our solution function and reflecting s_{zero} and s across the $s = 0$ line. So we make Transfun$(u_1, u_2, s_{zero}, G, s)$ physical by setting the u-value equal to u_1 in the region with $s < -s_max$, equal to SolFun$(u_2, u_1, -s_{zero}, G, -s)$ in the region with $s_{zero} > s > -s_{max}$, and equal to u_2 in the region with $s_{zero} < s$.

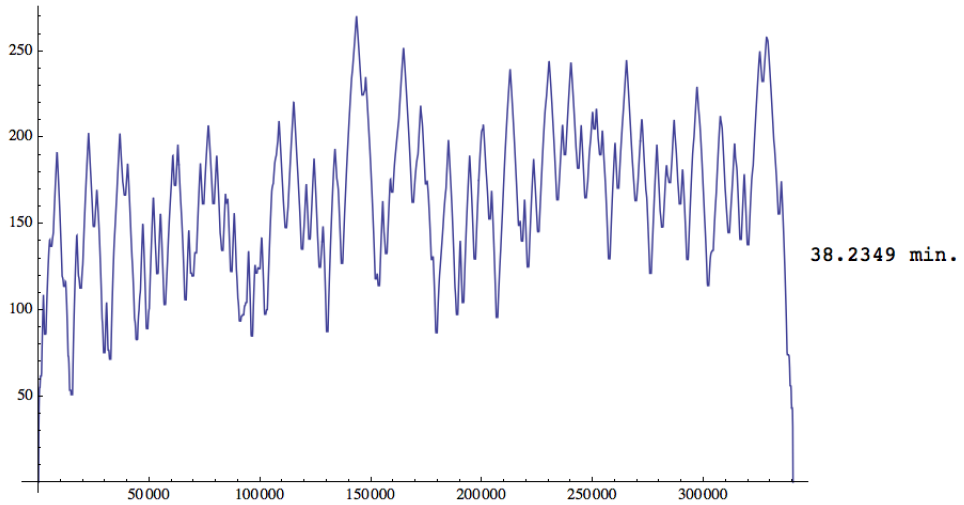
The next step is to define ConnectFun, which defines the u-value over a region with radius of curvature R and with boundary conditions u_1 and u_2. Given the myriad of detailed conditions for the piecewise functions to handle the following is a heuristic definition of ConnectFun. The initial step is to determine the forward transition function with boundary conditions u_1 and RG and the reverse transition function with boundary conditions RG and u_2. If the length of the region is sufficiently long that each transition function can attain its maximum constant value, then we simply join the transition functions over the appropriate domains. If however, the length of the region is not long enough for each transition function to attain its maximum constant value, then we set the u-value equal to the minimum of the 2 values which the distinct transition functions return.

1.4 Finding the Trip Time

Phew! We have finally solved for $v(s)$. Here is an example output for a driving route from San Francisco to Los Angeles under the .5 g limit:



And this output is for a driving route from New York to Boston:



Notice how the trip time is on the right of each image. How do we do this? Well, with our velocity profile $v(s)$ in hand, we can simply calculate the trip time by an integral, letting s range from 0 to S, where S is the length of the route:

$$\mathbf{T}(\boldsymbol{\gamma}) = \int_0^S \frac{ds}{v(s)} \qquad (dt = \tfrac{ds}{v})$$

Chapter 2

Calculating Money M($\boldsymbol{\gamma}$)

Implicit in space, separate from $\boldsymbol{\gamma}$ itself is the notion of the cost m as a function of location, $m : \mathbb{R}^2 \to \mathbb{R}$. Then

https : //writelatex.s3.amazonaws.com/hncypvvzcbqq/page/f82b09d3b9be4e73b3ed9050daffc

We throw out impossible locations by defining m piecewise:

$$m(x, y) = \begin{cases} \infty & \text{if impassible} \\ 20 & \text{if farm} \\ 5 & \text{if highway} \end{cases}$$

For cost we divided the various components into a set that depends solely on length $\{f(l)\}$

$$\text{Pylons} \tag{2.1}$$
$$\text{Tube construction} \tag{2.2}$$
$$\text{Linear Accelerators?} \tag{2.3}$$
$$\text{Solar Panels} \tag{2.4}$$

and a set that depends on position $\{f(\gamma(s))\}$

$$\text{Tunnel} \tag{2.5}$$
$$\text{Permits and Land} \tag{2.6}$$

2.1 Research

(a) Get response from Musk as to cost of individual components and land considerations

(b) Solve differential equation for v